CLAIMS

I claim :

1. A computer system or network comprising a local or distributed data or object caching system wherein:

(a) data or objects are (i) placed in a user cache in the caching system when the data or object is first read from a data source by a user as a lazy cache, or (ii) preloaded as directed by a user,

(b) each unit of data or object that is placed in a user cache is registered with a unit notification system (UNS) as being present in the cache and the UNS is connected to a Java Messaging Service (JMS) or to any other reliable messaging service available in the object programming language system environment,

(c) when any unit of data or object is updated in a user caches and persisted (saved) to a primary data source the UNS sends a message to all other user caches that may exist and contain the updated unit of data or object to register in the user cache a message that the unit was updated by another user and is noted as being invalid in the cache, and

(d) other user caches that have been notified as described in (c) will read from the data source the updated unit of data or object and update the cache only when a user attempts to access the invalidated unit or data or object in a user cache, may automatically update the user cache as soon as notified as has been pre-set in the caching system by the user of the user cache.

2. The computer system or network according to claim 1, comprising a local object or data caching system wherein multiple applications or users access the data in an object language virtual machine environment.

3. The computer system or network according to claim 2, comprising a distributed object

or data caching system that may be distributed over an intranet or internet environment and may include distributed users, distributed components, distributed caches, multiple cache copies, or any combination thereof wherein a central data source is used for insertions, deletions, or changes to the cached data or objects.

4. The computer system of claim 1, wherein the system also provides transparent persistence.

5. The computer system of claim 2, wherein the system also provides transparent persistence.

6. The computer system of claim 3, wherein the system also provides transparent persistence.

7. The computer system of claim 1, wherein the system further comprises at least one Object to Relational Mapping Repository.

8. A software module or modules for an object programming application comprising a local or distributed data or object caching component, or components, wherein the :

(a) the software component, or components, provide that data or objects are (i) placed in a user cache in the caching system when the data or object is first read from a data source by a user as a lazy cache, or (ii) preloaded as directed by a user,

(b) the software component, or components, provide the logic for placing each unit of data or object in a user cache and registering the unit or object with a unit notification system (UNS) as being present in the cache and the for connecting the UNS to a Java Messaging Service (JMS) or to any other reliable messaging service available in the object programming language system environment, and

(c) the software components, or components, containing logic providing that when any unit of data or object is updated in a user caches and persisted (saved) to a primary data source the UNS sends a message to all other user caches that may exist and contain the updated unit of data or object to register in the user cache a message that the unit was updated by another user and is noted as being invalid in their cache as an unreliable data unit or object.

9. The software module, or modules, of claim 8, wherein the software component, or components, further provide the logic directing other user caches that have been notified that a data unit or object has been updated to only read from the data source an updated unit of data or object to update their user cache when the invalidated unit of data or object in their user cache needs to be accessed, or provides that their cache may automatically be updated as soon as notified as has been pre-set in software by the user of the user cache.

10. The software module, or modules, according to claim 9, comprising logic to access a local object or data caching system wherein multiple applications or users access the data in an object language virtual machine environment.

11. The software module, or modules, according to claim 9, comprising logic to access a distributed object or data caching system that may be distributed over an intranet or internet environment and may include distributed users, distributed components, distributed caches, multiple cache copies, or any combination thereof wherein one or more central data sources are used for persisting changes to the cached data or objects.

12. The software module, or modules, of claim 8, wherein the software also provides transparent persistence.

13. The software module, or modules, of claim 9, wherein the software also provides transparent persistence.

14. The software module, or modules, of claim 10, wherein the software also provides transparent persistence.

15. The software module, or modules, of claim 11, wherein the software also provides transparent persistence.

16. The software module, or modules, of claim 9, wherein the software also provides a component or class for communicating with at least one Object to Relational Mapping Repository.